[19]中华人民共和国国家知识产权局

[51]Int.Cl[7]
G06F 9/45

[12]发明专利申请公开说明书

[21]申请号 00107022.3

[43]公开日 2000年11月1日

[11]公开号 CN 1271891A

[22]申请日 2000.4.24 [21]申请号 00107022.3

[30]优先权

[32]1999.4.23 [33]US [31]09/298251

[71]申请人 太阳微系统有限公司

地址 美国加利福尼亚州

[72]发明人 小C·N·克利克 C·A·维克

M·H·帕莱茨尼

[74]专利代理机构 中国专利代理(香港)有限公司

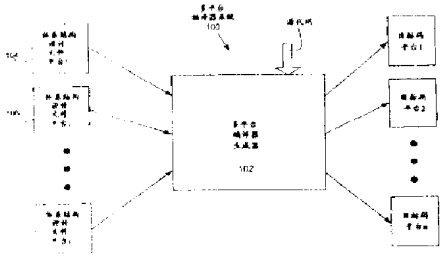
代理人 罗 朋 王忠忠

权利要求书3页 说明书13页 附图页数12页

[54]发明名称 用于在多平台环境的指令选择的装置和方法

[57]摘要

一种用于生成具有内嵌指令选择器的平台所特定编译器的系统和方法。一组用于描述一个依赖特殊硬件平台的编译器的体系结构特征的用户定义的依赖于平台的编译器的体系结构描述符和一组用于识别由指令选择器所选的平台所特定指令的指令判定被分别转换为依赖于平台的编译器目标码和指令选择器目标码。具有内嵌指令选择器的平台所特定编译器由依赖平台编译器目标码,指令选择器目标码和独立于平台的编译器目标码来形成。



ISSN1008-4274

权　利　要　求　书

1. 一种用于产生具有内嵌指令选择器的平台所特定编译器的装置，包括:

一组用户定义的依赖于平台的编译器的体系结构描述符，它用于描述一个依赖特殊硬件平台的编译器的体系结构特征;

一组用于识别由指令选择器所选的平台所特定指令的指令判定;

一个用于将用户定义的依赖平台编译器体系结构描述符转换为依赖于平台源代码和用于将该组指令判定转换为平台所特定指令选择器源代码的体系结构描述符编译器;

一个用于将依赖于平台的编译器源代码编译为依赖于平台编译器目标码和用于将平台所特定指令选择器源代码编译为内嵌的指令选择器目标码的主编译器;

独立于平台的编译器目标码; 和

一个在平台所特定编译器运作期间用作在依赖平台编译器目标码和独立于平台编译器目标码之间信息流的媒介的接口，其中在平台所特定编译器运作期间，内嵌指令选择器根据所选指令的执行成本来选择将要执行的指令，而其中内嵌的指令选择器提供隐式指令判定来由平台所特定编译器使用以编译所选择的指令。

2. 权利要求 1 的装置，其中，平台所特定编译器包括独立于平台编译器目标码和适合于在特殊硬件平台上执行的依赖于平台编译器目标码。

3. 权利要求 2 的装置，其中，在平台所特定编译器运行期间，独立于平台的编译器目标码通过提供一个信息请求给接口来请求一个依赖于特定平台的目标码信息，该接口将该信息请求直接引入依赖于预定平台的编译器目标码信息提取器。

4. 权利要求 3 的装置，其中，依赖于平台的编译器目标码信息提取器通过提取满足该信息请求的依赖于特定平台的编译器目标码来响应该信息请求。

5. 权利要求 4 的装置，其中，所提取的信息被提供给该接口，该接口依次将该信息引入信息请求器。

6. 权利要求 1 的装置，其中，所述装置包括多组用户定义的依赖于平台的体系结构描述符，其中，每组描述符相应于不同的硬件平台。

7. 一种用于生成具有内嵌指令选择器的平台所特定编译器的方法，包

括:

提供一组用户定义的依赖于平台的编译器的体系结构描述符，它用于描述一个依赖特殊硬件平台的编译器的体系结构特征;

提供一组用于由内嵌指令选择器使用来选择那些将在运行期间被平台所特定编译器编译的指令的用户定义的指令判定;

由一个体系结构描述符编译器来将该组用户定义的依赖平台的编译器体系结构描述符转换为依赖于平台的编译器源代码;

由一个体系结构描述符编译器将该组用户定义的指令判定转换为平台所特定指令选择器源代码;

由一个连接到体系结构描述符编译器的主编译器将依赖于平台的编译器源代码编译为依赖于平台编译器目标码;

由一个连接到体系结构描述符编译器的主编译器将指令选择器源代码编译为指令选择器目标码;

提供一个独立于平台的编译器目标码，其中，独立于平台的编译器目标码和依赖于独立于平台的编译器目标码适合于在特定硬件平台上执行;

由依赖平台编译器目标码，指令选择器目标码和独立于平台的编译器目标码来形成内嵌的指令选择器。

8. 权利要求 7 的方法，还包括:

在平台所特定编译器运行期间，由独立于平台的目标码由独立于平台的编译器目标码请求一个依赖于特定平台的目标码信息;

提供一个信息请求给该接口;

由该接口将该信息请求直接引入依赖于预定平台的编译器目标码信息提取器。

9. 权利要求 8 的方法，还包括:

为响应该信息请求，提取满足该信息请求的依赖于特定平台的编译器目标码。

10. 权利要求 9 的方法，还包括:

由该接口将所提取的信息引入该信息信息请求器。

11. 权利要求 10 的方法，还包括:

由该接口将所提取的信息引入该信息信息请求器。

12. 一个具有内嵌指令选择器的平台所特定的编译器，包括:

一个依赖于平台的编译器目标码;

一个独立于平台的编译器目标码，其中独立于平台的编译器目标码和依赖于平台的编译器目标码适合于在一个特殊硬件平台上执行；

一个嵌入依赖于平台编译器目标码中的依赖于平台的指令选择器目标码；

一个部分嵌入独立于平台代码和部分嵌入依赖于平台目标码中的接口，其中，在平台所特定编译器运行期间，接口作为在独立于平台编译器代码和依赖于平台编译器代码之间信息流的媒介。

13. 权利要求 12 的编译器，其中，在平台所特定编译器运行期间，独立于平台的编译器目标码通过提供一个信息请求给接口来请求一个依赖于特定平台的目标码信息，该接口将该信息请求直接引入依赖于预定平台的编译器目标码信息提取器。

14. 权利要求 13 的编译器，其中，依赖于平台的编译器目标码信息提取器通过提取满足该信息请求的依赖于特定平台的编译器目标码来响应该信息请求。

15. 权利要求 14 的编译器，其中，所提取的信息被提供给该接口，该接口依次将该信息引入信息请求器。

16. 权利要求 12 的编译器，其中，所述装置包括多组用户定义的依赖于平台的体系结构描述符，其中，每组描述符相应于不同的硬件平台。

说　明　书

用于在多平台环境的指令选择的装置和方法

本发明一般地涉及计算机系统。更具体地，涉及用于在多平台计算环境中选择可执行指令的方法和装置。

软件平台和硬件体系结构的连续增加确保了计算机用户和计算机程序开发员将要在他们的职业生涯中考虑许多不同的计算机环境。应注意的是，在本论述的上下文中，术语“环境”指在与移植（ported）软件交互的计算机系统中的元件的完整范围。这些元件典型地包括一个处理器和一个操作系统以及 I/O 设备，程序库，网络，或，在一些情况中的较大的人或物理系统。甚至在数准－标准平台（例如，IBM－PC，UNIX）已经广泛使用了，但仍然不是通用的计算机环境。为了维持和扩展它们的生存能力，因此，多数软件程序将最终面对被移植的要求，使得基于现有版本软件的可执行版本的软件程序在一个新的计算环境中产生。因此，可移植性，或一个软件程序被移植到一个给定环境（即，目标）的能力正在被普遍认识为一个对于多数软件程序而言的所希望的属性。因此，软件程序在不同计算平台之间的可移植性通过延长了它的生命周期和扩展了它所期望应用的的安装范围而明显增强了软件程序的价值。如在本领域所公知的，一个软件程序可以包括一个应用程序，一个系统程序，或一个程序的组件，相反一个软件系统是软件程序的集合。

如果一个软件系统的移植费用是和在某种程度上小于在新目标环境中重新编写程序的费用，则该软件程序是可移植的。如果一个软件程序可以零成本来移植，则它的可移植性是极佳的，当然这在现实中是不可能的。实际上，这由两种基本可移植性协议，第一种是二元可移植性（即，移植软件程序的可执行形式）和第二种为源可移植性（即，移植表示软件程序的源代码）。虽然，二元可移植性协议典型地提供了几个优点（主要与移植的容易度有关），但它只能用于将软件程序移植到非常相似的环境中，从而严重限制了它的可用性。相反，因为源可移植性协议采取了源代码的可用性，它们典型地提供了使一个特殊软件程序适用于更宽范围的计算环境的更大能力。

不幸地，多数移植过程仍然在使用 ad hoc 方法，这导致了低效率的技术，这实质上添加到了将一个软程序从一个平台移植到另一个平台上的成本。举例来说，一个编译器将一个计算机程序由一个语言翻译为另一个，在翻译过程中捕捉语法上的任何错误。最普通地，一个编译器将一些诸如 C+

+或 COBOL 的高级语言翻译为机器语言，使得计算机可以无需任何解释地理解。为了完全移植一个编译器，我们必须完成几个任务以使所移植的编译器可以成功地，和以一个高度可靠的方式，执行它所设计的功能，同时在一个和最初为它所构思的平台完全不同的平台上运行。

所要求完成以完全移植一个编译器的几个任务包括正确指令选择，因为通常许多不同的指令类型可以匹配相同的机器独立语义（machine independent semantic）。一个简单的例子是定义为将一个“1”的常数加到一个数值上的操作，其中“1”的值可以表示为一个 8 比特或 32 比特精确整数。在另一个例子中，对于一个在 Pentium®和 PentiumII®的微处理器系列中发现的 X86 处理器中，浮点单元（FPU）具有 3 个精确模式，其中它可以执行不同的操作，诸如加法和减法。在其中语义要求，例如，以 24 比特的精度来舍入，和一个 FPU 控制字已将 FPU 精度设为，例如，53 比特的情况中，对于 X86 编译器，选择在体系结构描述中定义的，产生一个 53 比特的精度而不引入附加舍入的指令是低效率和不正确的。

另外的例子包括多硬件平台，诸如生成为 V8 或 V9 处理器的 SPARC 微处理器。当生成为 V8 系统时，只有 V8 类型指令可被执行，然而，当生成为 V9 系统时，V8 和 V9 类型的指令都可被执行。因此，基本上，在其中 V8 系统运行的情况中，只能选择 V8 类型的指令，因为 V9 指令不能在 V8 系统中执行。

我们还希望不仅选择那些将在一个特殊平台上正确执行的指令，而且选择那些通过减少执行“成本”而改善处理器总性能的指令。例如，将诸如减法的一个特殊操作的结果保存在一个存储位置通常比将相同结果保存在一个数据寄存器中更加要求计算机资源（即，成本更高）。因此，成本最有效的方式，可能的话，是选择其成本在可选择的所有指令中最低的指令。使用上面的例子，我们由成本有效的观点可以体会到应选择与那些将它们各自结果保存在存储器中一个位置的指令相反的将其结果保存在一个寄存器中的指令。

因此，所希望的是定义一个选择协议的能力，从而不仅选择了正确的指令用以在多平台计算环境中执行，而且执行所选择的这些指令的成本最低。

广泛而言，本发明涉及一个用于生成一个具有在多平台环境中的指令选择器的编译器的改进方法，装置和计算机系统。本发明可以多种方式实现，包括一个方法，一个计算机系统和一个装置。下面将讨论本发明的几个实施例。

根据本发明的一个方面，我们希望有一个用于编译一个具有内嵌指令选择器的平台特定编译器的装置。该装置包括一组用户定义的依赖平台的编译器体系结构描述符和指令选择所使用的指令判定（instruction predicates），该描述符描述了一个特殊硬件平台的相应的体系结构特征。一个体系结构描述符编译器将用户定义的依赖平台的编译器体系结构描述符转换为一个依赖平台的编译器源代码和被一个主编译器转换为依赖平台的目标码与指令选择器目标码的指令选择器源。该编译器由独立于平台的编译器目标码和依赖平台的编译器目标码结合指令选择器目标码来组成。

如用于生成一个平台特定编译器的方法，它提供了一组用于描述一个依赖特定硬件平台的编译器的相应体系结构特征的用户定义的依赖平台的编译器体系结构描述符和指令判定。该描述符和指令判定被一个体系结构描述符编译器转换为依赖平台的编译器源代码和指令选择器源代码。该依赖平台的编译器源代码和指令选择器源代码被编译为依赖平台的目标码和指令选择器目标码。具有内嵌的指令选择器的平台特定编译器由依赖于平台的目标码，指令选择器代码和独立于平台的编译器目标码构成。

在另一个实施例中，公开了一个平台特定编译器。该编译器包括一个具有内嵌的指令选择器目标码和适于在特殊硬件平台上执行的独立于平台的编译器目标码的依赖于平台的编译器目标码。一个部分嵌入该独立于平台的代码和部分嵌入依赖于平台的目标码的接口在平台特定编译器运行期间作为在独立于平台的编译器代码和依赖于平台的编译器代码之间的媒介。

本发明的这些和其他优点将在阅读了下面的详细描述和对各个附图进行研究之后变得明显。

本发明以及其进一步的优点可以参照下面的描述结合附图得到最好的理解。

图 1 是根据本发明的一个实施例的多平台编译器系统的表示性框图；

图 2 显示了图 1 中多平台编译器的特定实例；

图 3 显示了图 2 所示的编译器的另一实例；

图 4 详细显示了根据本发明一个实施例的编译器生成过程的流程图；

图 5 显示了具有根据本发明一个实施例的平台特定编译器的 Java 虚拟机（JVM）；

图 6A 显示了一个根据本发明一个实施例的 AD 文件组织结构；

图 6B 显示了在包括在图 6A 所示 AD 文件中的各种数据域之间的特殊关

系；

图 7 显示了一个根据本发明一个实施例，连接依赖于平台的源代码和独立于平台的源代码的示例性接口；

图 8 是一个根据本发明的由编译引擎进行的运行过程的示例性表示；

图 9 是一个独立于机器的指令的表示；

图 10 是一个详细描述根据本发明一个实施例的指令选择过程的流程图；

图 11 显示了一个用于实施本发明的计算机系统。

在下面的描述中，将描述在多平台计算环境中选择指令的构架和方法。本发明最初以驻留在 Java 虚拟机中的多平台编译器的形式进行描述。一般地，为了生成一个具有内嵌的指令选择代码，公开了一组以一个体系结构描述语言（ADL）文件形式的用户定义的平台特定体系结构描述符和一组用户定义的指令判定。应注意地是，ADL 可以采取对于本领域技术人员公知的许多形式，诸如 C++，属性文法，习惯描述语言等，或这些形式的一些组合。在所述实施例中，该指令判定采取与嵌入平台特定编译器的指令选择代码性结合布林（boolean）表达式的形式，使得只有这些指令将被执行。在本发明的一个实施例中，对布林表达式值的确定只能在指令选择过程进行。

AD 文件包括用户定义的指令判定，这构成到包括一个体系结构设计语言编译器（ADLC）的多平台编译器的输入。在一个实施例中，ADLC 用于产生用于生成目标特定编译器的依赖于特殊目标平台的编译器源代码。ADLC 还产生目标平台指令选择源代码和目标平台指令判定源代码。具有内嵌指令选择器的平台特定编译器是使用由 ADLC 提供的依赖于平台的编译器源代码，指令选择源代码和指令判定源代码结合另外提供的独立于平台的编译器目标码来生成。

以此方式，平台特定编译器不仅选择那些由用户确信适合于在目标平台上执行的指令，还选择那些通过减少指令执行成本而改善处理器性能的指令。以此方式，在多平台环境中处理器的可靠性和性能都明显改善。

图 1 是根据本发明的一个实施例的多平台编译器系统 100 的表示性框图。该多平台编译器系统 100 可以生成，或编译一个可以选择适合于目标平台的指令的平台特定编译器。另外，编译器还可以选择那些那些通过减少指令执行成本而改善处理器性能的指令。

在所述实施例中，平台特定编译器是通过编译表示平台特定编译器与任何特殊平台无关的那些部分的独立于平台源代码和表示编译器平台特定的那

些特征的依赖于平台源代码来生成的。在其中提供用户定义（显式地）的指令判定，或其中它确定特殊指令要求判定而不管它们是否由用户提供（隐式地）的情况中，编译器是使用提供具有上述选择能力的平台特定编译器的指令判定源代码来生成的。

更具体地，在所述实施例中，多平台编译器系统 100 包括一个被配置用于使用平台特定体系结构描述符和指令判定来生成依赖于平台的编译器的编译器生成器 102。在所述实施例中，平台特定体系结构描述符采取用于表示目标平台编译器依赖于平台的那些部分的体系结构描述符语言（ADL）的形式，同时指令判定采取布林表达式的形式。应注意的是，ADL 可以采取许多对于本领域技术人员公知的形式，诸如 C + +，Java 源代码，Pascal 等。典型地，它可以是由多平台编译器系统 100 的用户编码的 AD 文件，然而在一些情况中，AD 文件在被称为“转键（turn key）”系统的情况中是由原始设备制造商提供。在这些情况中，终端用户已选择了被确信是有用的目标平台而提供商已提供了必需的译码努力。

在许多情形中，AD 文件位于，例如，在编译器生成器 102 的存储器系统中，诸如 AD 文件 104 和 AD 文件 106。应注意地是，虽然可以提供任何数量的 AD 文件，每一个特定于一个特殊平台，然而，在一个时间只有一个 AD 文件被编译器生成器 102 处理。以此方式，一个平台类型 1 被提供，当特定于，例如，平台类型 1 的依赖于平台的源代码被提供时，定制的编译器生成器 102 可以将源代码编译为平台类型 1 的目标码。例如，文件 104 可以包括具有被编译器生成器 102 用于生成可以择优地选择那些由用户定义的指令判定使能的指令的 X86 编译器的指令判定的 ADL 代码。如在本领域公知的，X86 目标码是那些由一个 X86 微处理器可执行的指令，X86 微处理器是由 Santa Clara，Ca 的 Intel 公司所制造的。

沿相同脉络，AD 文件 106 可以包括具有被编译器生成器 102 用于生成可以择优地选择那些由用户定义的指令判定使能的指令的 SPARC 编译器的指令判定的 ADL 代码。

在两种情况中，处理器总的性能可以得到改善，这是因为一个成本分析还可以与选择同时进行。以此方式，编译器生成器 102 可以自动提供与 AD 文件和可用的用户定义指令判定同样多的具有适当指令选择代码的平台所特定编译器。

现在参照显示了图 1 所示多平台编译器生成器 102 的特殊实例的图 2. 在所示实施例中，编译器生成器 102 包括一个连接到一个具有用户定义的指令判定文件 201 的 AD 文件 204 的体系结构描述符语言编译器 202（ADLC）。该 ADLC202 被配置来分别将包含在 AD 文件 204 中的平台所特定 ADL 代码和包含在文件 201 中的指令判定编译为依赖于平台的编译器源代码，指令选择器源代码和指令判定源代码。另外，ADLC 202 可以产生不是显式写入 AD 文件 204 中的指令判定以确保一个正确或有效的指令选择，在这里被称为隐式指令判定。

依赖于平台的编译器源代码，指令选择器源代码和指令判定源代码被依次提供给一个连接到 ADLC 202 的主编译器 203。在所述实施例中，主编译器 203 是一个本领域公知的 C++ 编译器。然而，任何适于将源代码编译到目标特定编译器目标码的编译器都可以使用。

在编译过程中，连接到 ADLC 202 的主编译器 203 将依赖于平台的源代码结合依赖于平台的指令判定源代码和指令选择器源代码编译以形成具有内嵌的指令选择目标码的依赖于平台的编译器目标码。在所述实施例中，依赖于平台的编译器目标码由包括内嵌的指令选择器目标码的代码块 206 来表示，该指令选择器目标码由包含在连接于主编译器 203 的编译器单元 208 中的代码块 216 来表示。内嵌的指令选择目标码提供选择那些被用户确信为适当的指令的能力给习惯编译器单元 208。另外，通过执行一个成本分析，那些减少指令执行成本的指令可以被择优地选择，从而通过减少指令执行的总成本来改善处理器性能。

在一些情况中，独立于平台的编译器目标码可由主编译器 203 所编译的独立于平台的编译器源代码得到。在其他情况中，诸如所述的实施例，独立于平台的编译器目标码已被提供。在所述实施例中，独立于平台的目标码由一个在一个实施例中被包含在连接于一个独立于平台接口的编译器引擎 212 中的代码块 210 来表示。在一个特殊实例中，在所谓运行时间的过程中，接口 214 作为将在依赖于平台的目标码和指令选择目标码之间信息传送到编译引擎的媒介。

举例来说，当要求一个 X86 微处理器时，对于该例子，用户提供一个在此已保存了对于 X86 平台所特定的适当 ADL 代码的 AD 文件 204 和具有适当指令判定的文件 201。通过编译器生成器 102 的引导，X86 特定的 ADL 代码和指令判定都被提供给 ADLC 202。然后，ADLC 202 将 ADL 代码编译为 X86

所特定的编译器源代码和将指令判定编译为相应的平台所特定的指令判定源代码和指令选择器源代码。应注意的是，ADLC 202 是一个可以将任何合适结构 AD 文件转换为相应平台所特定的编译器源代码，和如果要求，转换为依赖于平台的指令判定源代码的通用编译器。以此方式，基本上消除了用户译码，或以任何方式修改 ADLC 202 或它的任何部分的任何要求。因此，对用户的唯一译码要求是要求提供一个合适结构和验证过的可以包括，如果需要，指令判定的 AD 文件。

一旦 ADLC 202 已将 X86 所特定的 ADL 代码编译为 X86 编译器源代码和将指令判定编译为指令选择器源代码，它们被编译为使用主编译器的目标码和分别由块 206 和 216 来表示。

在一些例子中，如图 3 所示，依赖于多平台的编译器源代码文件对于编译器单元 300 是可用的。编译器单元 300 是图 2 所示的编译器单元 208 的一个实例，因此自然只能被认为是示例性的。在所述实施例中，各种依赖于平台的 AD 文件被保存在文件栈 301 中。文件栈 301 对于编译器单元 300 可以在本地的，或在一些情况中，可以位于，例如，数据基址，远端服务器等的远端。这种配置特别适于包括在诸如 Internet（互联网），局域网（LAN）等类似的计算机网络上传输数据的应用中。多 AD 文件的使用特别有利，这是因为它提供了如相应 AD 文件表示的任何平台所特定的编译器所需要的操作能力。举例来说，文件栈 301 包括一个 AD 文件 302 代表，例如，SPARC 特定描述符随着它们相应的指令判定的平台，该指令判定被转换为对应的依赖于平台的源代码 304 和指令选择器 306。通过此安排，任何具有其对应的包含在文件栈 301 中的 AD 文件的平台可以被选择来定制该编译器单元 300。虽然未示出，一个选择器典型地被用于选择被输入到 ADLC202 的 AD 文件。

如上所讨论的，编译器生成器 102 生成了一个具有指令选择器的特殊编译器，在一个实施例中使用如图 4 中流程图所详细描述的过程 400。该过程 400 通过提供以 ADL 代码形式保存在，例如，一个 AD 文件中的平台所特定的体系结构描述符而在 402 处开始。在 404 处，它确定该 AD 文件是否包括用户所定义的指令判定（显式的）或是否要求隐式的指令判定。当，例如，对于正确的指令执行而言特定的数据基本上在特定的位置时，则将要求隐式的指令判定。这一情况是随着具有一个特定操作数的多次出现的那些指令而发生的，其中每一操作数必须是相同的，否则可能出现一个操作错误。因为错误的风险可能随着不同平台而变化（即，在一个 2 地址平台上执行一个 3 地

址指令相对于在一个 3 地址平台上执行一个 2 地址指令），依赖用户去理解这一事实是危险的。另外，一个指令判定可能要求编译器单元的实现细节的使用，该细节可以对于 AD 文件的作者是不可用的。因此，隐式指令判定是指令选择过程的一部分，其中那些要求特殊考虑的特殊指令由 ADLC 提供了适当的指令判定。

当要求显式和/或隐式指令判定时，ADLC 在 406 提供指令选择器源代码。ADLC 基本与 406 同步地，在 408 处基于 AD 文件输入提供平台所特定编译器源代码，从而，独立于平台的编译器源代码在 410 被提供。然后，主编译器通过在 412 同时编译独立于平台的源代码，依赖于平台的编译器源代码和依赖于平台的指令判定源代码来生成具有指令选择和执行成本分析能力的平台所特定的编译器。

一旦在 414 已生成了平台所特定的编译器，则平台所特定的编译器在运行期间对于选择那些被编译为所需的平台所特定目标码的成本有效指令而言是可用的。在一个特定实施例中，在运行期间，独立于平台的接口是作为在编译引擎和依赖于平台的编译器源代码和指令判定源代码之间的信息流的媒介。

更近地，Java 编程语言，一个面向对象语言，已引入可以编译可以在配置由 Java 虚拟机（或字节码解释器）的任何计算机系统平台上运行的输出（称为字节码）的能力。Java 虚拟机被设计来将字节码解释为可以由实际的硬件处理器执行的指令。字节码利用这种虚拟机，而非一次被解释一个指令，可以在每一特定平台上由，在一些情况中，实时（just-in-time）编译器重编译。

图 5 显示了一个包括一个 Java 虚拟机（JVM）500 的装置，该虚拟机包含在根据本发明一个实施例的编译器单元 208 中。在所述配置中，连接到 ADLC 202 的平台所特定的 AD 文件栈 502 包括一组 AD 文件，其中每一个 AD 文件代表依赖于特定平台的编译器特征。在所述实施例中，AD 文件栈 502 包括分别代表 X86 和 SPARC 体系结构的 AD 文件 104 和 AD 文件 106。在其中一些不同 AD 文件包含在 AD 文件栈 502 的情形中，选择器单元 502（未示出）典型地被用于由相应于所需操作平台的 AD 文件栈 502 来选择一个特定 AD 文件。当选择了一个适当的 AD 文件时，ADLC 202 将包含所余选 AD 文件中的 ADL 代码转换为如上所述的依赖于适当平台的编译器源代码。

在 Java 编程语言和环境中，一个实时（JIT）编译器是将 Java 字节码

变换为可以直接发送给处理器的指令的程序。在已编写了一个 Java 程序后，Java 编译器将 Java 源程序语句编译为 Java 字节码，而非编译为包含匹配一个特定硬件平台处理器（例如，一个 Intel Pentium 微处理器或 IBM 系统/390 处理器）的指令的代码。该 Java 字节码是可以发送给任何平台和在该平台上运行的独立于平台的代码。

更具体地，当字节码被提供给由编译器单元 208 所提供的 JIT 编译器时，包含在字节码 504 中程序的编译被延迟直至该程序将要运行。当字节码 504 被提供给一个解释器 506 时，字节码 504 被每次一个字节码地读入解释器 506 中。然后，解释器 506 在每一字节码被读入解释器 506 时，执行由每一字节码所定义的操作。也即，解释器 506“解释”字节码 504，这是本领域技术人员所公知的。通常，解释器 506 处理字节码 504 并基本连续地执行与字节码 504 有关的操作。

当一个程序被编译时，编译器单元 208 产生由一个指令选择器 508 所选择的机器指令。然后，编译器单元 208 由所选择的字节码 504 来产生机器指令，而所得到的机器语言指令可以直接由目标平台操作系统 510 来执行。通常，当虚拟机 500 终止时，机器语言指令被丢弃。

现在参照图 6，它显示了根据本发明一个实施例的 AD 文件 600 的组织结构。应注意的是，所示组织结构是 AD 文件 104 可以采取的许多可能的结构中的一个。在所示实施例中，AD 文件 600 是分层构造的一组不同平台体系结构描述符数据域。举例来说，一个寄存器定义数据域 602 被 ADLC 用于描述单个寄存器和具有目标体系结构的各类寄存器。一个编码块数据域 604 将由目标编译器所使用的编码类指定为输出字节流。一个帧管理块数据域 606 包括定义帧结构和管理协议的信息。这种信息包括，例如，帧栈增加的方向，由监视器进入操作所消耗的栈片数目，栈的对准要求，为“栈顶”保留的栈片数目，和其他。

一个操作数数据域 608 提供必须在指令定义之前用于在编译中正确编译的操作数定义，这是因为操作数构成用于指令定义中的用户定义的类型。一个管线（pipeline）规则数据域 610 被提供用于定义目标体系结构管线的行为。一个指令定义数据域 612 提供用于目标体系结构的指令格式以及相应的指令判定。一个窥孔（peephole）数据域 614 提供由 ADLC 所使用的目标体系结构所特定的优化规则。

AD 文件 600 的分层结构强调在各种 AD 文件数据域直接的相互关系。图

6B 以曲线显示了这样一个关系，具体地，是在包含于指令定义数据域 612 中的各个操作数之间的关系。通过执行一个由使用作为根域的指令定义数据域 612 进行的后向遍历（back traversal），对于各个操作数而言要求输入到 AD 输入数据域中的相关性可以被确定。例如，通过执行一个起始于作为根部的指令定义数据域 612 并沿各个分支延伸的正向遍历，管线规则数据域 610 和操作数定义数据域 608 被遭遇。执行一个起始于管线规则数据域 612 的正向遍历是遭遇寄存器定义数据域 602，同时，当执行一个起始于操作数定义数据域 608 的正向遍历时，将遭遇寄存器定义数据域 602 和编码类数据域 608。以此方式，为完全定义一个特定指令定义的各个操作数被提供。

图 7 示出了一个根据本发明，由依赖于平台的编译器使用的示例性接口 700。该接口 700 作为在编译器引擎 212（或独立于平台的目标码）与在块 206 中的依赖于平台的目标码之间的媒介。在所示实施例中，ADLC 输出包括用于确定性有限自动机（DFA）702 的代码，它指定了由理想操作数到机器指令的映射。该 ADLC 输出还包括定义了一组用于，例如，定义合法注册表征码，编码方法，分支偏移行为等的指令类 704 的目标码。窥孔规则 oracle760 的，用于指定合法的机器所特定树以优化的目标码和对于那些树的正确替换也由 ADLC 输出。以此方式，平台所特定的体系结构的特征被自动配置于一个适合于编译引擎 212 的格式中以在将源代码编译为依赖于平台的目标码的过程中使用。

接口 700 的独立于目标的部分被连接到一个匹配器（mathcer）708，该匹配器产生一个将被 DFA702 处理的输入树，该 DFA 执行颠倒重写规则系统（BURS）式样树模式匹配以选择用于在中间表示中的理想操作数的机器指令。

接口 700 还包括用作执行树模式匹配以发现最佳候选者和用最佳的机器指令树来替换所匹配的机器指令树的目标码配置。一个匹配器 708 使用匹配器 DFA 来执行指令选择和生成机器特定的中间表示。在注册分配器 712 选择一个合法的注册赋值给在机器特定表示中的操作数的同时，一个调度程序 710 排列机器所特定的中间表示。这包括将数值重新分配给正确位置（诸如，将变元移动到它们由调用约定所指定的正确位置）所需的任何指令的插入。一个窥孔优化器 714 模式匹配在机器所特定表示中的小树并用更优化的机器特定树来替换它们。一个目标码输出 716 使用虚拟调用来对诸如在缓存中的机器目标码的机器所特定表示进行编码，并使缓存对于虚拟机可用。

图 8 是一个在根据本发明一个实施例的编译机 212 的运行期中执行的指令编译过程的示例性显示。在运行期间，编译机 212，当要求时，产生一个对于来自依赖于平台的编译器目标码的特定信息和，如果需要，指令选择器目标码的信息请求（参考，在一个特殊实施例中，是发射功能调用）。举例来说，当编译机 212 要求平台所特定信息以使用内嵌指令选择器来选择一个指令，该编译机 212 执行一个对信息请求器 802 的功能调用。该信息请求器 802 依次连接到将该功能调用直接引入连接到具有所要求指令判定的 DFA 的信息提取器 804 的接口。在所述实施例中，DFA 然后基于相应的指令判定选择指令。该信息提取器 804，依次，通过提取所要求信息来响应该功能调用。应注意的是，所提取的信息以一种容易被编译机 212 使用的方式来构造。

图 9 表示根据本发明一个实施例的，一个具有一个操作数多次重现的指令的独立于机器的示例性曲线 900。该独立于机器的曲线 900 与那些具有相同语义的依赖机器指令相映射。例如，曲线 900 的节点 902 表示在最基本层次上的可以映射到任何数目的具有与减法相应的相同语义的依赖机器减法操作（subA, subB, subC）的减法操作。然而，如上所述，不是所有可能的映射都是所希望的，这是因为一些映射实际上可能导致错误。这可能，例如，产生在那些其中目标平台为 2 址处理器（诸如 X86），和减法操作的结果被要求保存在一个与输入不同的位置（正如在 SPARC 体系结构中 3 址指令所做的）的情况中。典型的，一个颠倒匹配过程用于将表示机器指令的二元树击穿为目标平台可以理解和执行的语义。在该例子中，该树的每一节点被映射到由指令判定使用 ADLC 所产生的指令选择代码来选择的依赖平台的指令。

根指令节点 902 指示该指令 900 为一个具有作为节点 904 和 906 输入结果的整数减法操作。该减法操作的结果在 908 处被保存在一个寄存器中。在此例中，指令选择器选择成本最有效的指令或所有匹配将要执行的指令的指令输出序列.

图 10 是一个详细描述根据本发明一个实施例的指令选择过程 1000 的流程图。过程 1000 通过在 1002 查找一个在独立于机器的指令表示二元树中的节点来开始。在 1004 处，相应于所定位的节点的指令选择代码被获取和执行。然后，在 1006 检查可能匹配所定位节点的下一个潜在用户所定义的指令。该匹配是基于在节点的独立于机器的表示和该节点的目标平台表示之间的语义的匹配比较的。在 1008，判定用户定义指令的输入位置是否匹配所定位的节点的输入位置。如果确定它们是匹配的，则在 1010 处判定该指定判

定是否满足。如果该指令不匹配或该指令判定不满足，则控制返回 1006，在此选择匹配该节点的下一个用户定义指令。

回到 1010，如果指令判定满足，则在 1012 对用户所定义的指令进行执行成本的估计。在 1014 处判定该估计成本是否小于前一估计成本。如果成本估计为小，则在 1016 处刷新成本估计变量并在 1018 处刷新该最佳指令，其后在 1020 处判定是否具有更多的用户定义的匹配指令。如果没有匹配指令，则过程停止，然而，如果这有更多的匹配指令，则控制返回 1006。

回到 1014，如果判定成本估计不小于前一成本估计，则如果这有另外的匹配指令，则控制返回 1006，否则过程停止。

图 11 显示了一个用于实现本发明的计算机系统 1100。该计算机系统 1100 或，更具体的，CPU1102 可以被配置用以支持虚拟机，这是本领域技术人员所公知的。如本领域所公知的，ROM 用作将数据和指令单向地传输到 CPU1102，同时 RAM 典型地用作以双向方式传输数据。CPU1102 通常可以包括任何数目的处理器。主存储设备 1104，1106 都可以包括任何合适的计算机可读媒介。次存储设备媒介 1108，它通常为一个海量存储器，也双向连接到 CPU1102 和提供附加的数据存储能力。该海量存储器 1108 是一个可用于保存包括计算机代码，数据等的计算机可读媒介。典型地，海量存储设备 1108 是一个诸如硬盘或磁带的通常比主存储设备慢的存储媒介。海量存储设备 1108 可以采取磁的或读纸带机或其他公知的形式。我们希望保存在海量存储设备 1108 中的信息可以，在适当例子中，被作为 RAM 1106 的一部分的虚拟存储器包含在标准式样中。一个诸如 CD-ROM 的特定主存储设备 1104 还可以将数据单向传到 CPU 1102。

CPU 1102 还连接到一个或多个输入/输出设备 1110，该设备可以包括，但不限于，诸如视频监视器，轨迹球，滑鼠，键盘，麦克风，触摸式显示器，传感器读卡器，磁或纸带读带机，写字板，输入笔，声音或手写识别器，或其他公知的输入设备，诸如当然包括其他计算机。最后，CPU 1102，可选地，可以使用一个通常如 1112 所示的网络连接连接到一个计算机或通信网络上，例如一个互联（Internet）网络或一个企业内部（intranet）网络。通过这种网络连接，我们可以预期 CPU 1102 在执行上述方法步骤过程中可以由该网络接收信息，或将信息输出到该网络。这种信息，经常被表示为一序列将使用 CPU 来执行的指令，可以，例如以嵌入到载波中的计算机数据信号的形式由网络提取或输出到网络。上述设备和材料对于在计算机硬件和软件领域

的技术人员而言是熟知的。

虽然只公开了本发明很少的几个实施例，但应理解，本发明可以许多其他形式来实现而不背离本发明的范围。举例来说，多平台编译器可以用于任何计算系统。

虽然根据本发明的，将编译器由一个操作系统移植到另一个，不同操作系统的方法特别适合于对应于基于 Java™的环境来实现，但这些方法通常可以适用于任何合适的基于对象的环境。特殊的是，这些方法适合于在独立于平台的基于对象的环境中使用。应理解的是，该方法还可以在一些分布式面向对象的系统中实现。

在使用一个分布式基于对象的计算机系统来描述本发明的同时，应理解的是，本发明一般可以在具有编译器的任何合适的计算系统中实现。因此，前面的描述应被人为是示例性的和非限制性的，而本发明不限于这里所给出的细节，可以在权利要求的范围内修改而具有完全等效的范围。

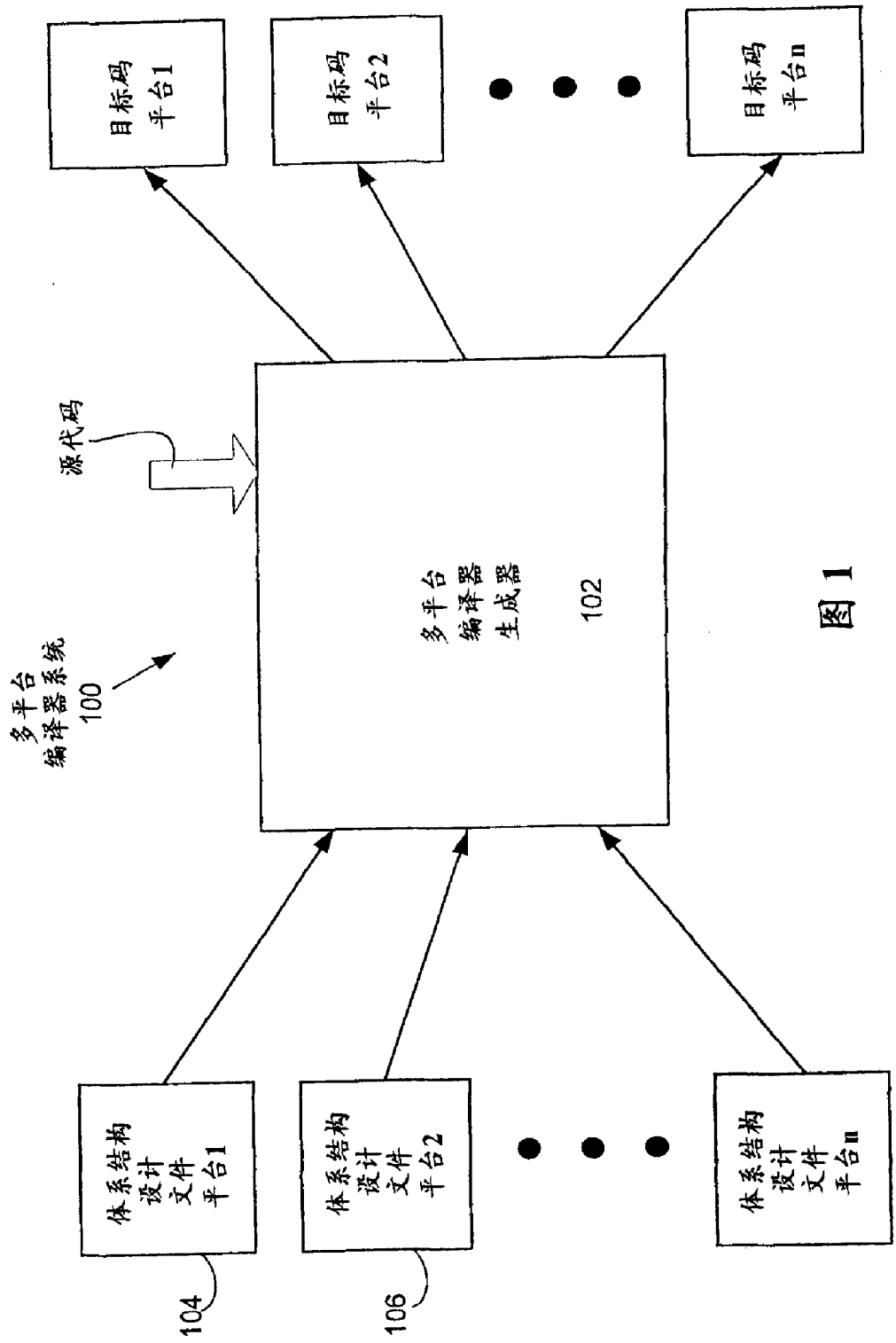

目标码
平台1
目标码
平台2
目标码
平台n
源代码
多平台
编译器
生成器
102
多平台
编译器系统
100
体系结构
设计
文件
平台1
体系结构
设计
文件
平台2
体系结构
设计
文件
平台n
104
106

图 1

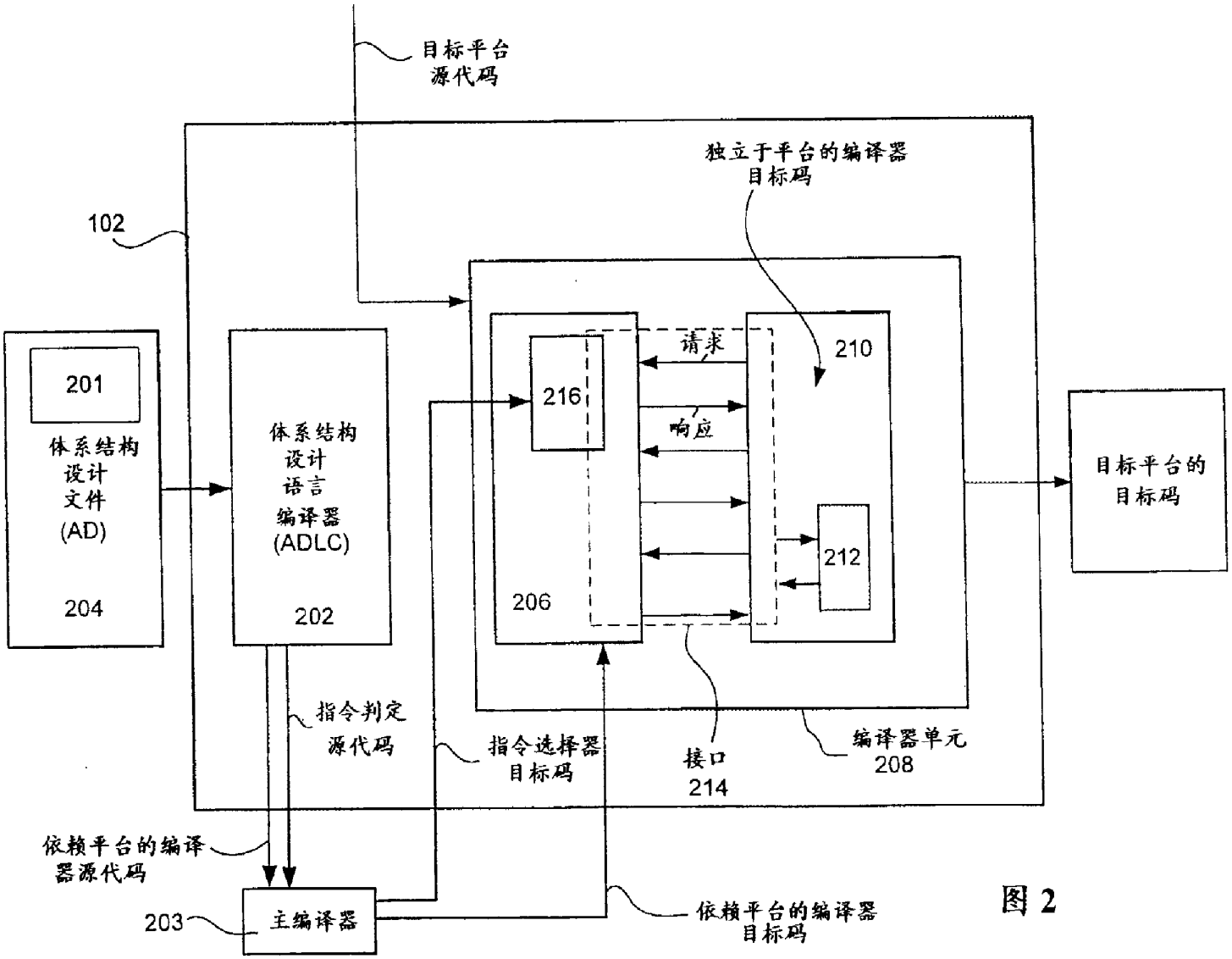

目标平台
源代码
独立于平台的编译器
目标码
102
201
体系结构
设计
文件
(AD)
204
体系结构
设计
语言
编译器
(ADLC)
202
216
请求
响应
210
212
206
目标平台的
目标码
指令判定
源代码
指令选择器
目标码
接口
214
编译器单元
208
依赖平台的编译
器源代码
203
主编译器
依赖平台的编译器
目标码

图 2

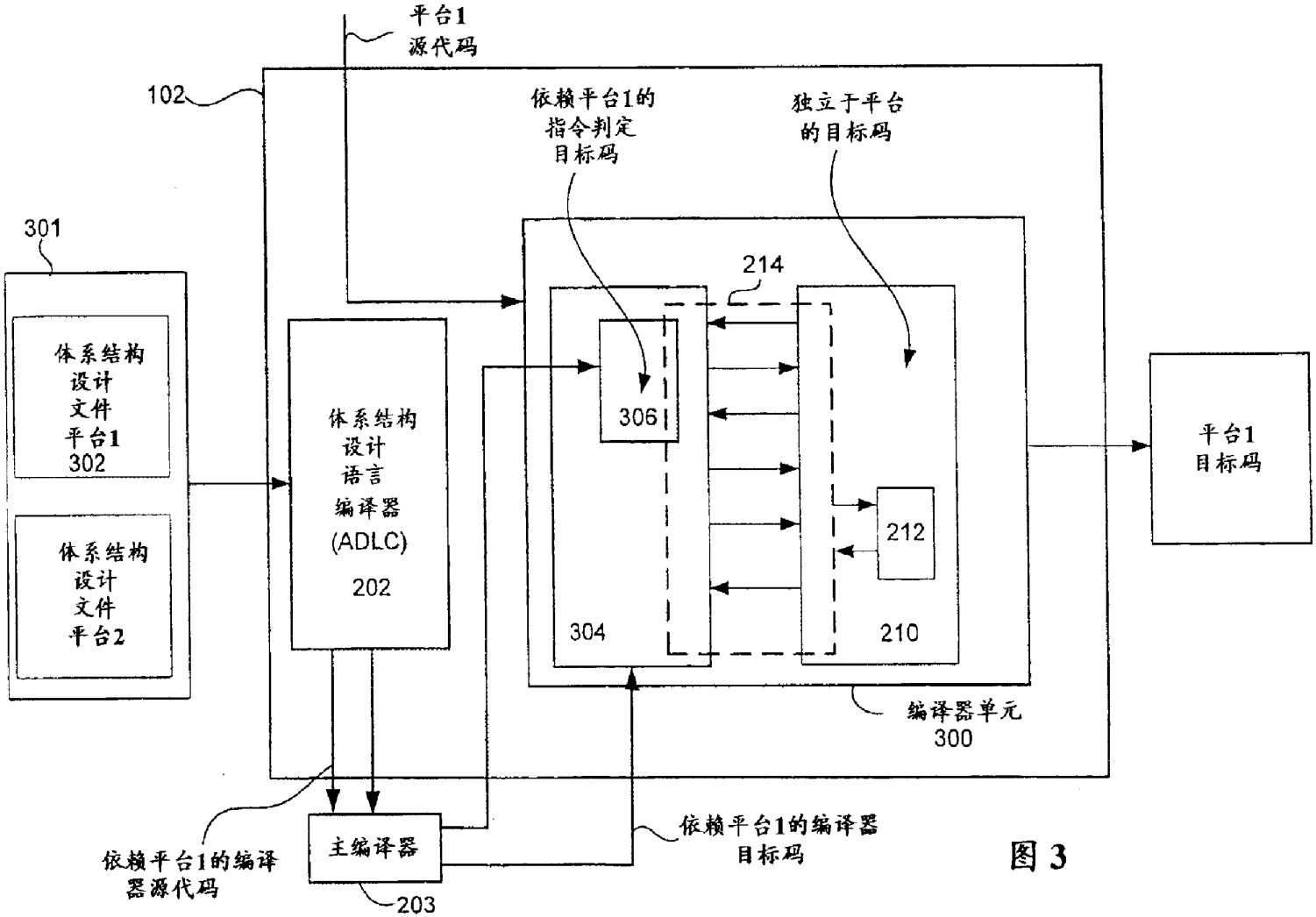

平台1
源代码
102
依赖平台1的
指令判定
目标码
独立于平台
的目标码
301
214
体系结构
设计
文件
平台1
302
体系结构
设计
文件
平台2
体系结构
设计
语言
编译器
(ADLC)
202
306
212
304
210
编译器单元
300
平台1
目标码
依赖平台1的编译
器源代码
主编译器
203
依赖平台1的编译器
目标码

图 3

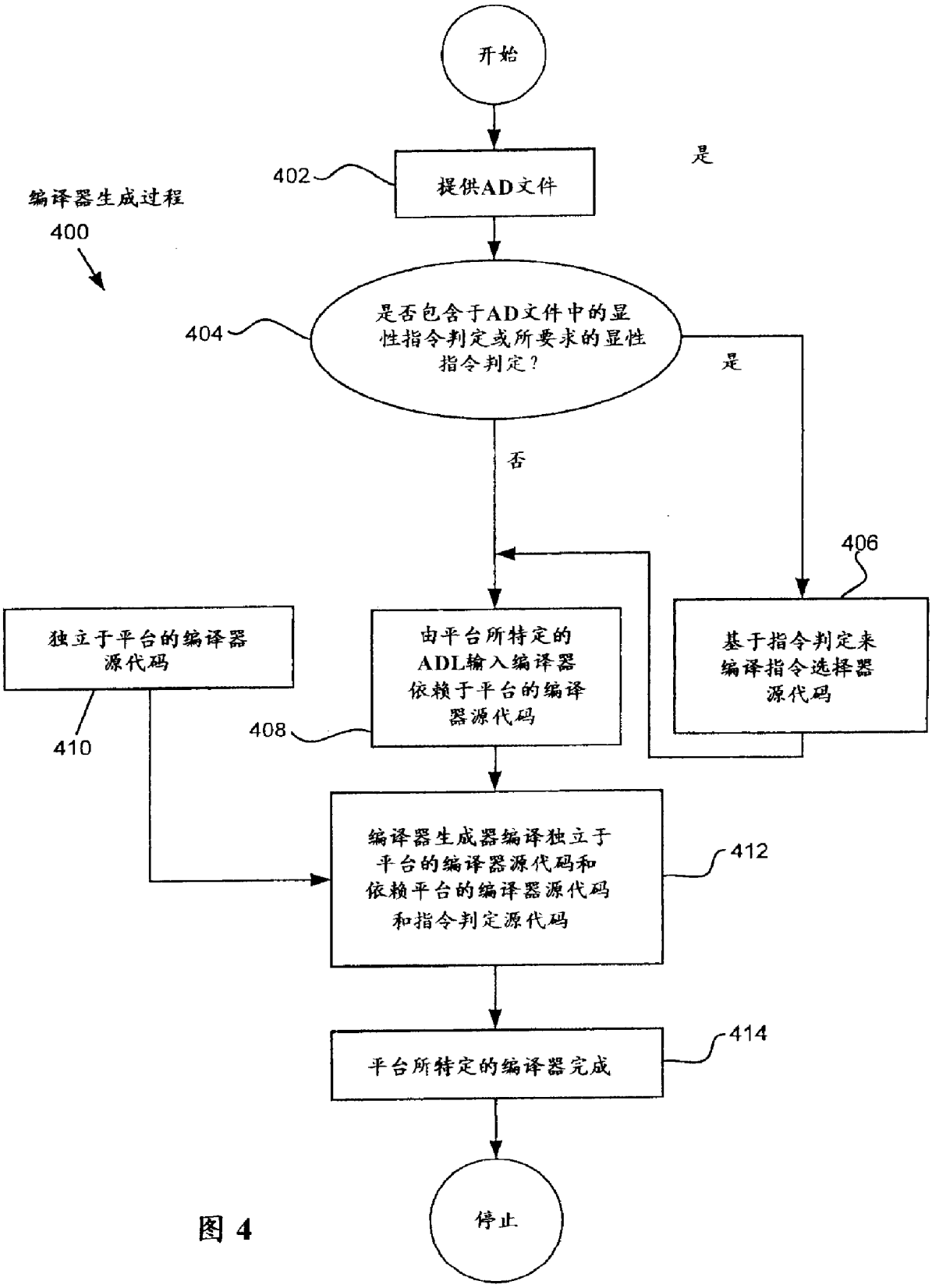

编译器生成过程
400
开始
402
提供AD文件
是
404
是否包含于AD文件中的显性指令判定或所要求的显性指令判定？
是
否
406
基于指令判定来编译指令选择器源代码
独立于平台的编译器源代码
410
由平台所特定的ADL输入编译器依赖于平台的编译器源代码
408
编译器生成器编译独立于平台的编译器源代码和依赖平台的编译器源代码和指令判定源代码
412
平台所特定的编译器完成
414
停止

图 4

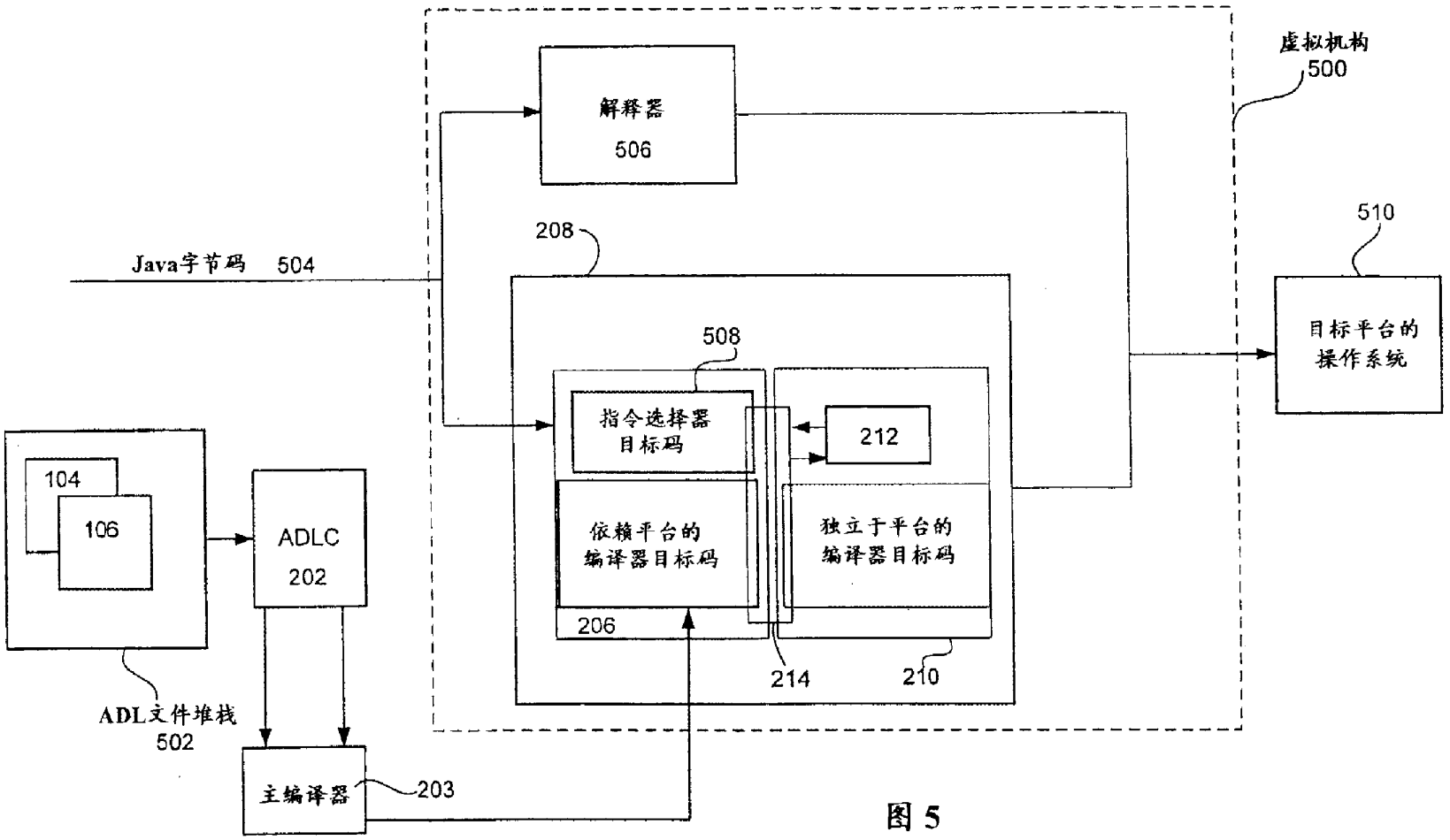

虚拟机构
500
解释器
506
510
208
Java字节码 504
目标平台的
操作系统
508
指令选择器
目标码
212
104
106
ADLC
202
依赖平台的
编译器目标码
独立于平台的
编译器目标码
206
214
210
ADL文件堆栈
502
主编译器
203

图 5

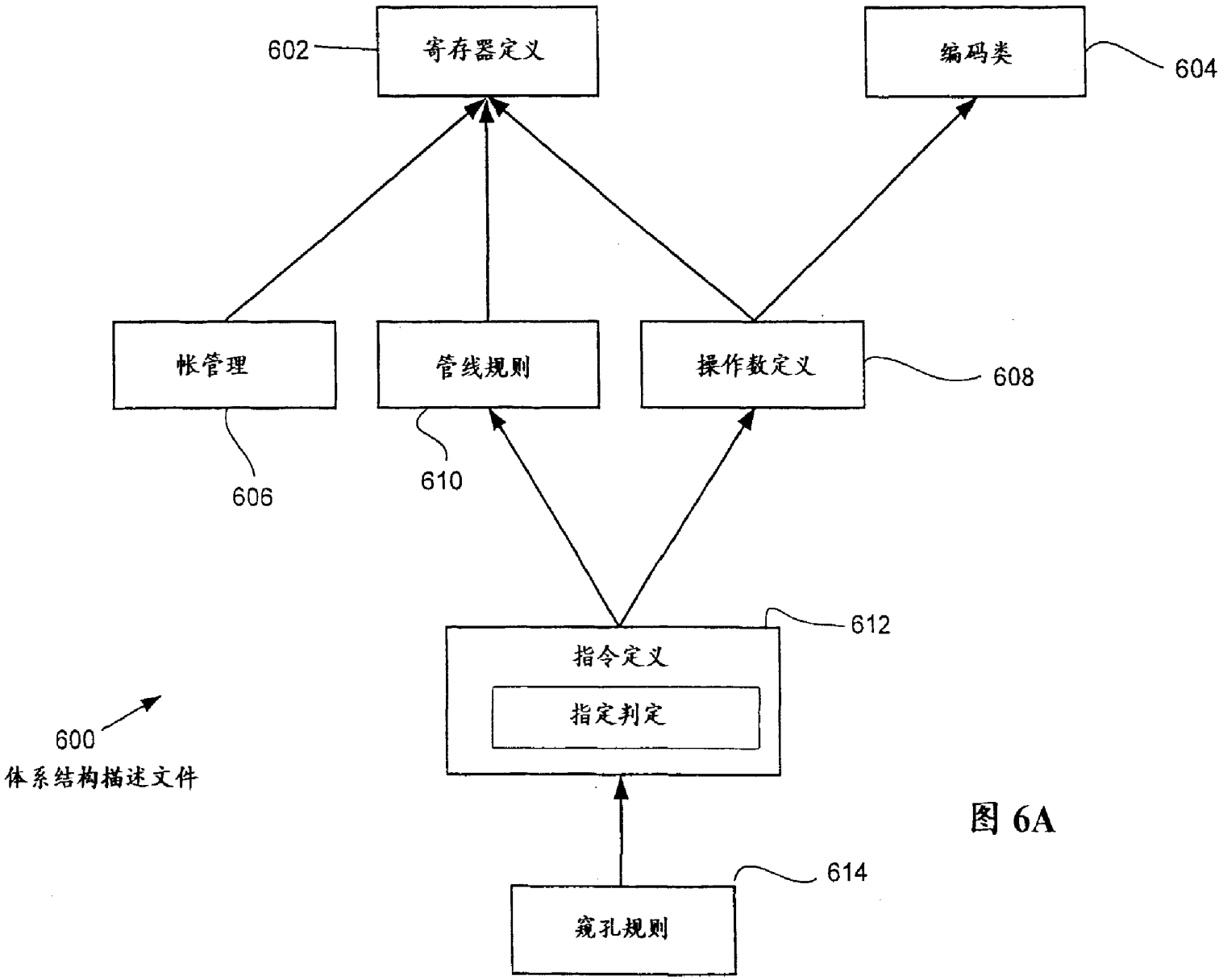

602
寄存器定义
编码类
604
帐管理
管线规则
操作数定义
608
606
610
612
指令定义
指定判定
600
体系结构描述文件
614
窥孔规则

图 6A

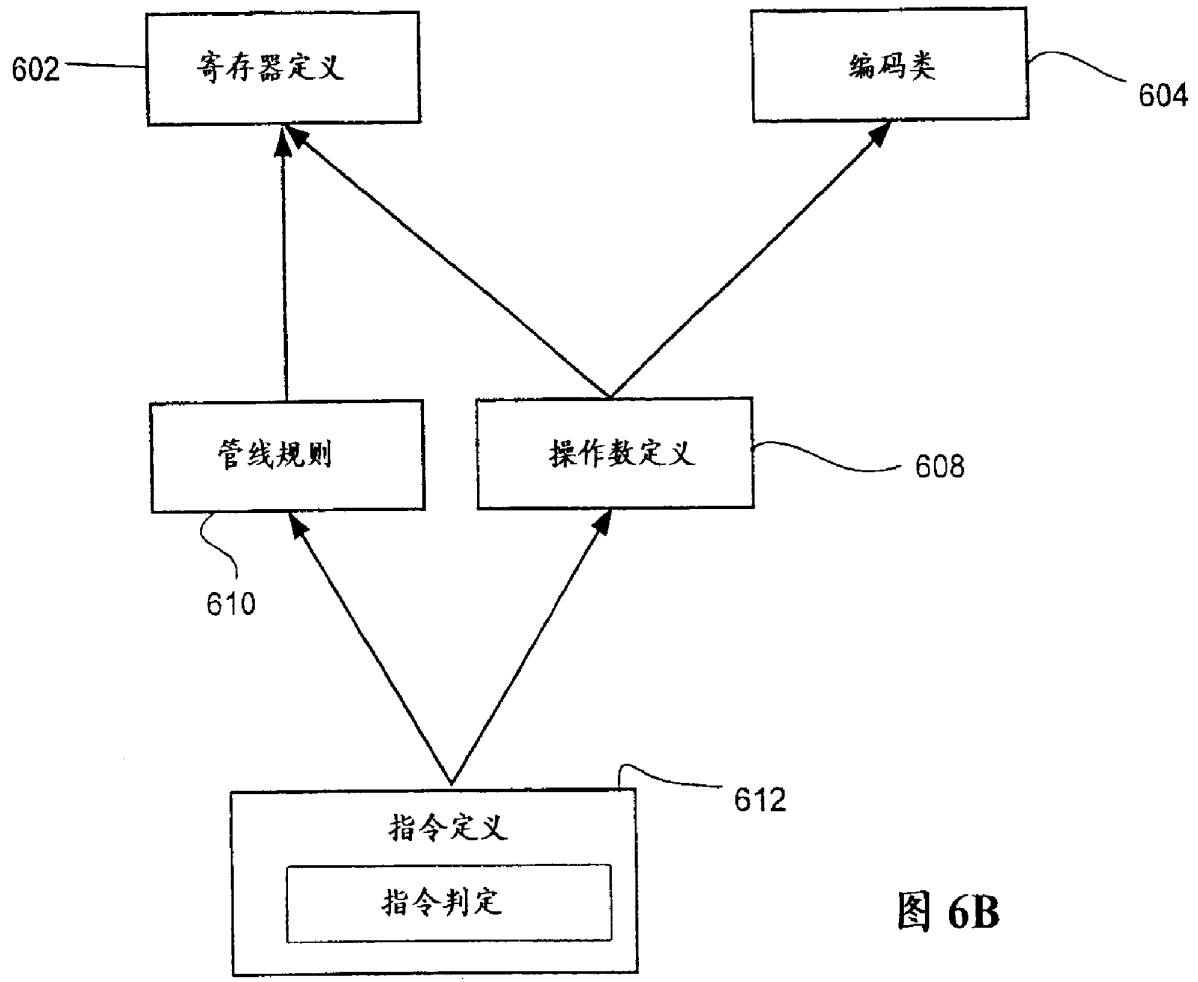


图 6B

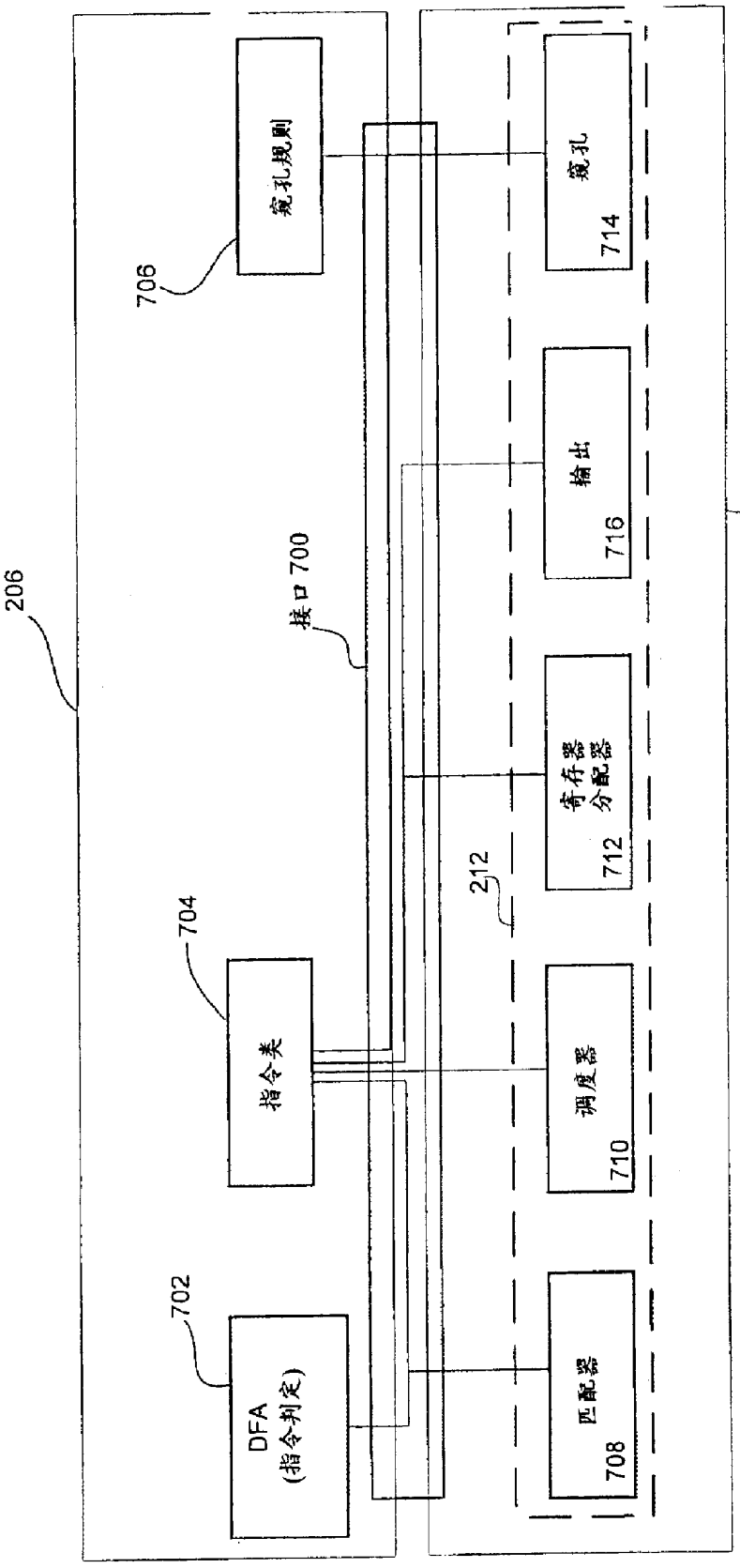

206
210
706
窥孔规则
接口 700
212
704
指令类
702
DFA
(指令判定)
匹配器
708
调度器
710
寄存器
分配器
712
输出
716
窥孔
714

图 7

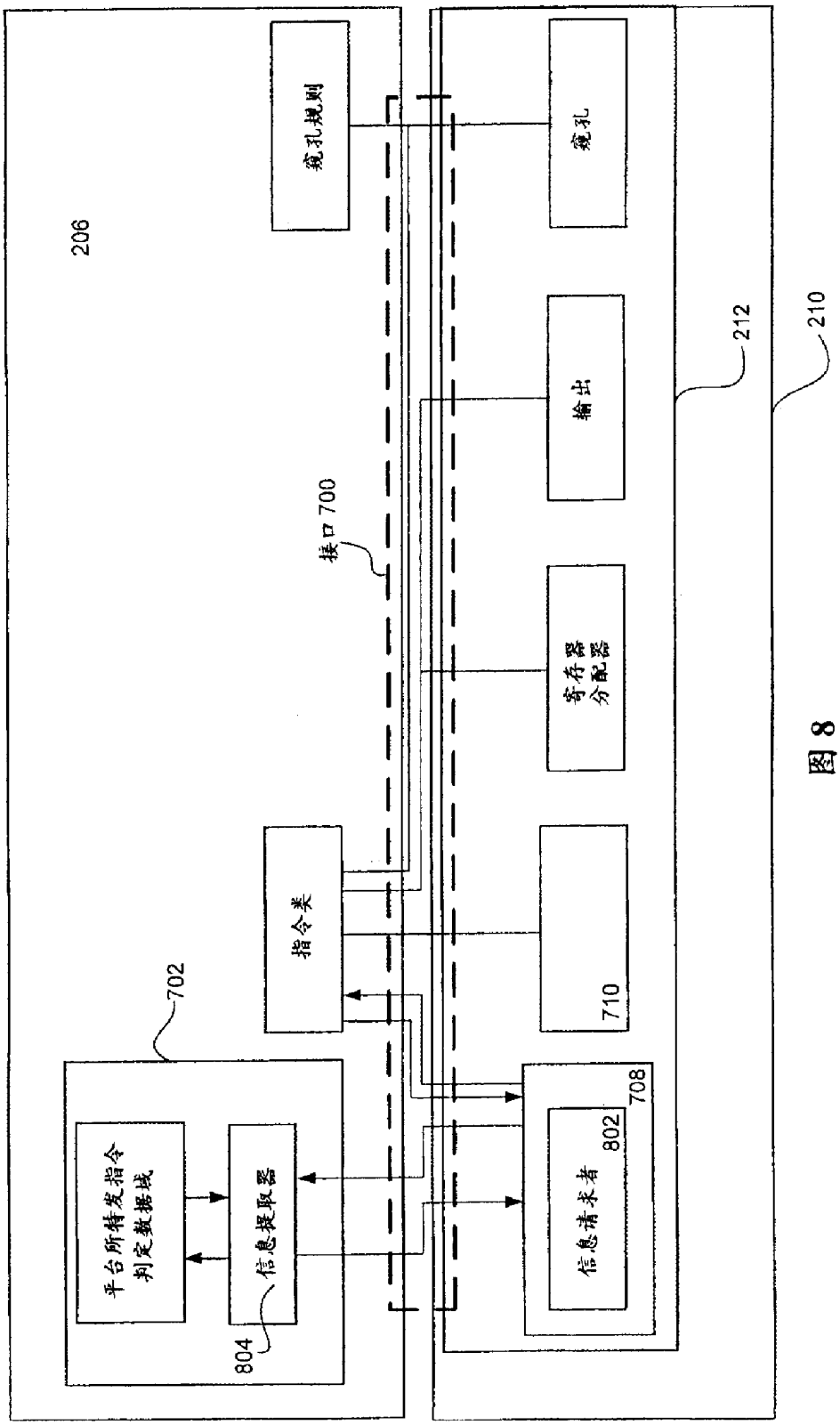

206
窥孔规则
窥孔
输出
212
210
接口 700
寄存器
分配器
指令类
710
702
平台所特发指令
判定数据域
信息提取器
804
708
信息请求者
802

图 8

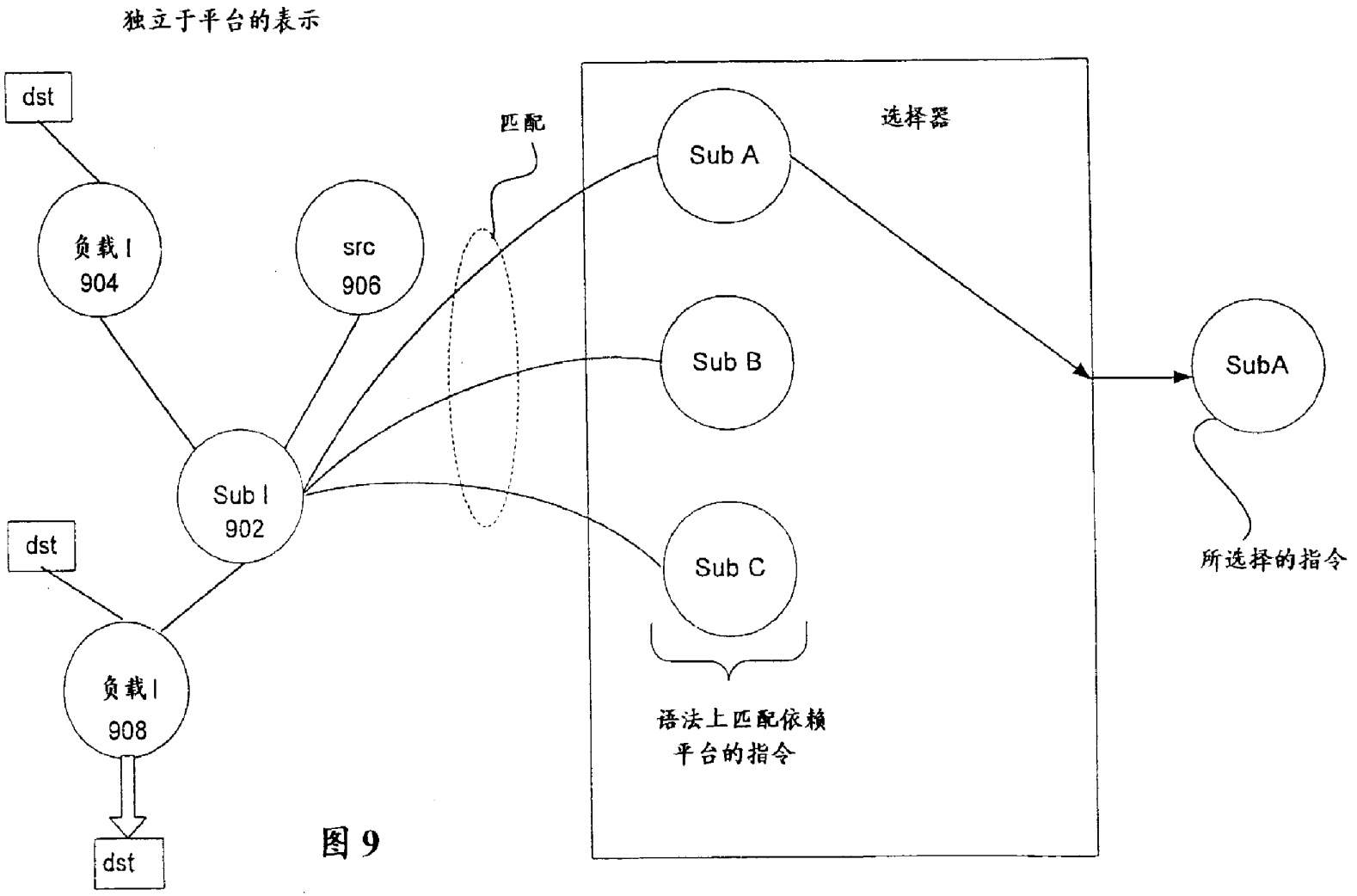

独立于平台的表示
dst
负载 I
904
src
906
匹配
选择器
Sub A
Sub B
Sub C
SubA
Sub I
902
dst
负载 I
908
dst
所选择的指令
语法上匹配依赖
平台的指令

图 9

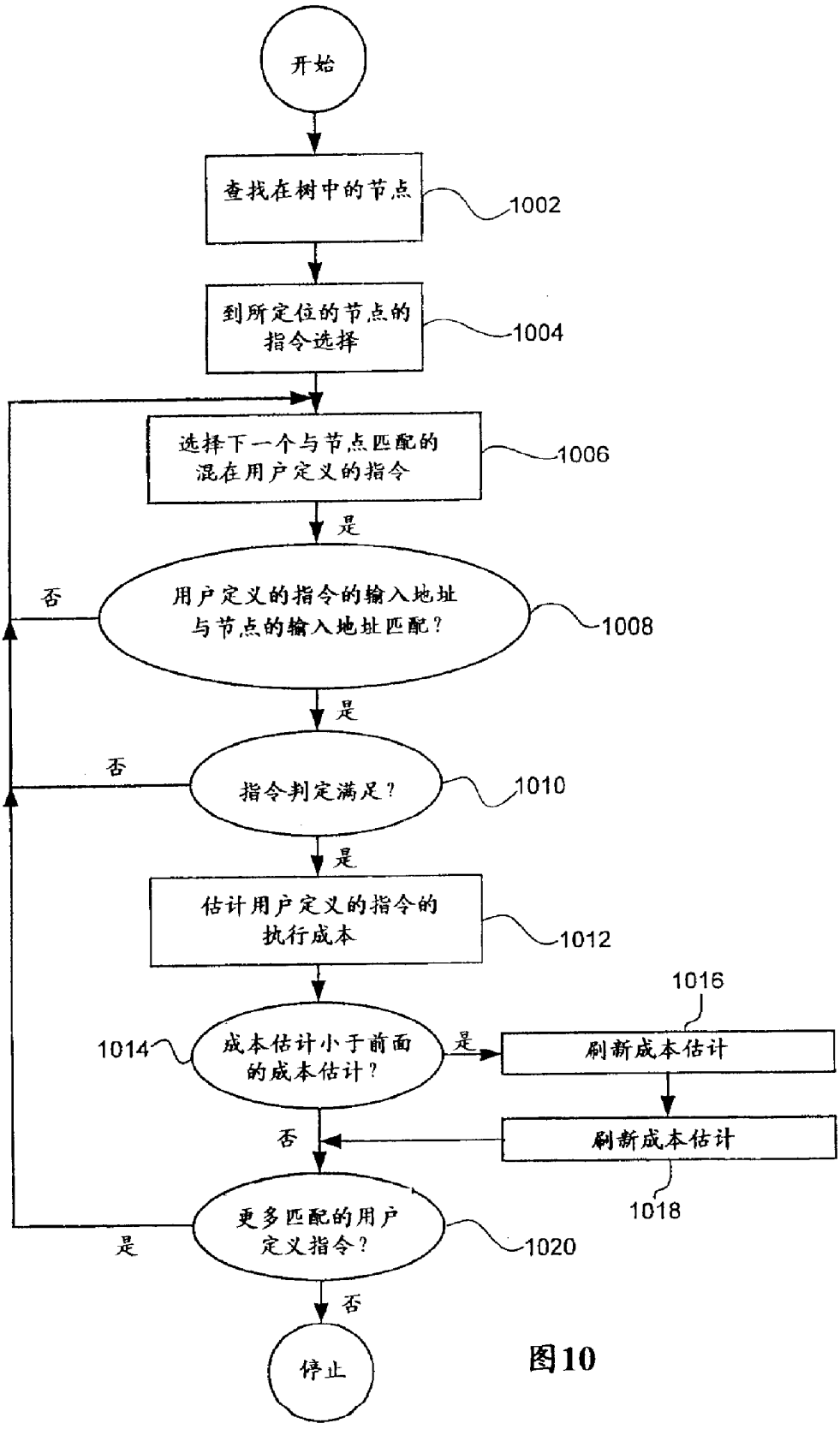

开始
查找在树中的节点
1002
到所定位的节点的指令选择
1004
选择下一个与节点匹配的混在用户定义的指令
1006
是
否
用户定义的指令的输入地址与节点的输入地址匹配？
1008
是
否
指令判定满足？
1010
是
估计用户定义的指令的执行成本
1012
1016
1014
成本估计小于前面的成本估计？
是
刷新成本估计
否
刷新成本估计
1018
更多匹配的用户定义指令？
1020
是
否
停止

图10

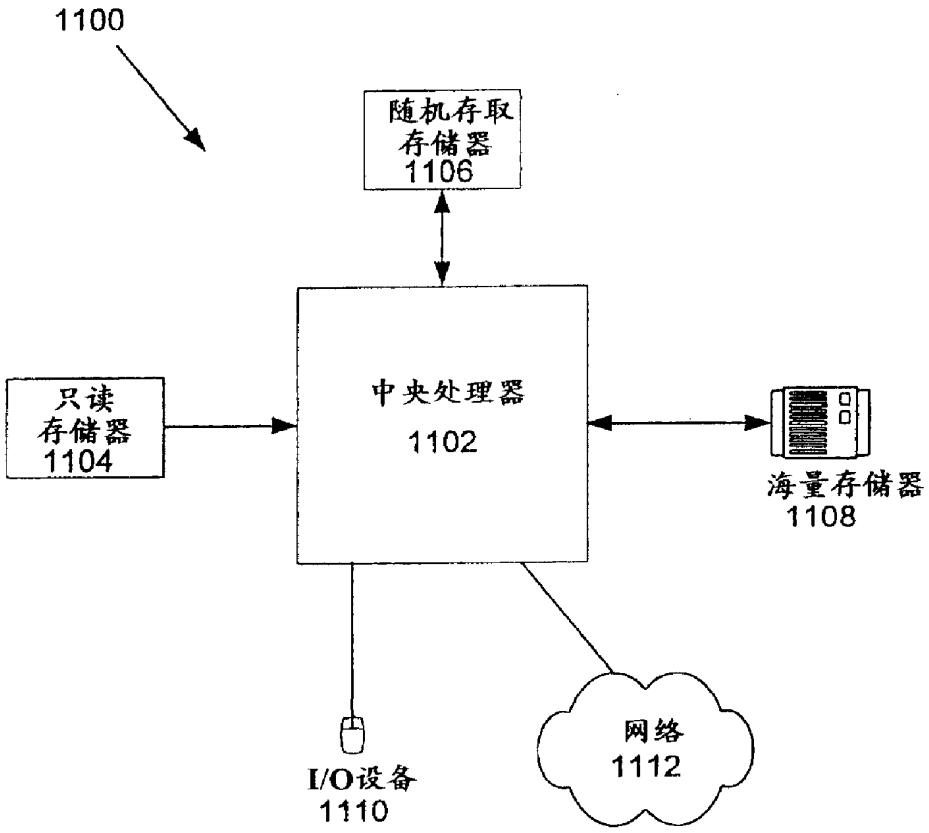

1100
随机存取
存储器
1106
只读
存储器
1104
中央处理器
1102
海量存储器
1108
I/O设备
1110
网络
1112

图 11